实验五　计数器和时钟

2020 年秋季学期

杨飞洋 191220138

目录

1. 实验目的：

- 复习计数器的工作原理，通过学习几种简单计数器的工作过程和设计方法、以及开发板系统时钟的使用，学习计数器的设计和定时器的工作原理。
- 学习 FPGA 开发平台上时钟源的使用，并结合计数器的设计方法学习定时器的设计
- 在 DE-10 Standard 开发板上实现一个电子时钟，时钟要求能够显示时、分、秒；还可以有以下功能：调整时间；闹铃（在特定时间 LED 闪烁）；秒表；等

2. 实验原理（知识背景）

利用触发器可以构成简单的计数器，通过级联可以构造多位的计数器。
在计数器的时钟输入端输入一个固定周期的时钟，那么计数器就变成了定时器。

3. 实验环境/器材

系统：windows10
开发软件：Quartus 17.1 Lite
开发板：DE10 Standard
仿真环境：ModelSim
芯片：Cyclone V ， 5CSXFC6D6

4.设计思路

总体来说就是先把各个功能的模块分别实现好，再在总的模块中连线调用。

先完成两个基础模块，分别是分频器和 7 段数码管显示器。比较容易，代码如下：

分频器模块：

```
module divider
#(
    parameter n
)
(
    input clk,
    output reg my_signal = 0
);
    integer clk_count = 0;

    always @ (posedge clk) begin
        if(clk_count == 25000000*n) begin
            clk_count <= 0;
            my_signal <= ~my_signal;
        end
        else
            clk_count <= clk_count + 1;
    end

endmodule
```

7 段数码管显示模块：

```
module trsltr(number,s);
    input [3:0]number;
    output reg [6:0]s;

    always @(*)
        case(number)
            0:s = 7'b1000000;
            1:s = 7'b1111001;
            2:s = 7'b0100100;
            3:s = 7'b0110000;
            4:s = 7'b0011001;
            5:s = 7'b0010010;
            6:s = 7'b0000010;
            7:s = 7'b1111000;
            8:s = 7'b0000000;
            9:s = 7'b0010000;
            default:s = 7'b1111111;
        endcase

endmodule
```

接下来开始设计电子时钟，需要实现正常工作、调时间、秒表、闹钟四个功能。
总模块中的输入接口：

```
module my_clock(
    input clk,
    input stop,
    input sw,
    input starp,
    input clr,
    input alarm,
    input adjust_h,
    input adjust_m,
    input adjust_s,
    output [6:0]hour1,
    output [6:0]hour2,
    output [6:0]minute1,
    output [6:0]minute2,
    output [6:0]second1,
    output [6:0]second2,
    output bell
);
```

clk 输入 50MHz 的时钟信号，stop 是调时间选项，sw 是秒表功能选项，starp 是秒表的开始暂停键，clr 为秒表清零键，alarm 为闹钟功能选项，adjust_h、adjust_m、adjust_s 为调整时间的按键。输出的 hour1 为小时的十位数，hour2 为小时的个位数，以此类推另外两个。

时钟信号：

首先先输出一个周期为 1s 的时钟信号，可以用分频器输出，只要将分频器的参数置为 1 即可。，如下图：（signal 为目标时钟信号）

```
wire signal;
divider#(1) d(clk,signal);
```

接下来这个信号就可以输送给各个模块了。

输出时间：

一共有三套时间需要显示，分别是真实时间、秒表时间、闹钟时间，之所以比功能个数少了一套，是因为调时间和正常工作可以共用一套时间，就是说在用户调时间的时候，时钟是不正常运作的，而在进行秒表操作和闹钟设置的时候，是要正常运作的。

我在总模块中用了三套 wire 变量来存储三套时间，如下图

```
wire [4:0]hour;wire [5:0]minute;wire [5:0]second;
wire [4:0]hour_sw;wire [5:0]minute_sw;wire [5:0]second_sw;
wire [4:0]hour_al;wire [5:0]minute_al;wire [5:0]second_al;
```

当然还需要一套时间来存储输出到屏幕的时间，即：

```
wire [4:0]hour_display;wire [5:0]minute_display;wire [5:0]second_display;
```

还需要一个选择端来决定显示哪套时间，可以用两个选择键 sw 和 alarm
sw = 1, alarm = 0 ： 进行秒表操作，并且显示秒表时间
sw = 0, alarm = 1 ： 进行闹钟设置，并且显示闹钟时间
sw = 0, alarm = 0　or　sw = alarm = 1：显示正常时间

输出显示器模块部分代码：（省略了输入输出的声明部分）

```
always @(posedge signal) begin
   if(sw &&~alarm)begin
      hour_display <= hour_sw;
      minute_display <= minute_sw;
      second_display <= second_sw;
   end
   else if(~sw &&alarm)begin
      hour_display <= hour_al;
      minute_display <= minute_al;
      second_display <= second_al;
   end
   else begin
      hour_display <= hour_true;
      minute_display <= minute_true;
      second_display <= second_true;
   end
end
```

接下来谈一下各个模块的实现方法

时钟正常行走/调整时间模块

输入输出：

```
module move(signal,stop,adjust_h,adjust_m,adjust_s,hour,minute,second);
   input signal;
   input stop;
   input adjust_h;
   input adjust_m;
   input adjust_s;
   output reg [4:0]hour = 5'b00000;
   output reg [5:0]minute = 6'b000000;
   output reg [5:0]second = 6'b000000;
```

signal 为时钟信号，stop 为 0 时，正常行走，stop 为 1 时，调整时间。
当 stop 为 1，adjust_h 为 1 时，hour 向前调，遇 23 变 0，以此类推

正常行走的代码类似三个计数器的级联，逻辑比较简单，下图为逻辑代码

```
if(second == 59) begin
   second <= 0;
      if(minute == 59) begin
         minute <= 0;
         if(hour == 23)
            hour <= 0;
         else hour <= hour + 1;
      end
   else minute <= minute + 1;
end
else
   second <= second + 1;
```

调整时间的逻辑也比较简单，就是按住就往前走，不按就停下，下图为逻辑代码：

```
if(adjust_h) begin
   if(hour == 23)
      hour <= 0;
   else hour <= hour + 1;
end

if(adjust_m) begin
   if(minute == 59)
      minute <= 0;
   else minute <= minute + 1;
end

if(adjust_s) begin
   if(second == 59)
      second <= 0;
   else second <= second + 1;
end
```

这就完成了整个模块的编写，只需要将总模块中的时钟信号接入，stop 接入，adjust_h，adjust_m，adjust_s 位接入即可，下图是总模块接口：

```
move m(signal,stop,~adjust_h,~adjust_m,~adjust_s,hour,minute,second);
```

取反的原因是调时间的位在板子上连的是 button 键

秒表模块：

逻辑和正常行走差不多，总的来说还是三个计数器的级联，三个计数器分别对应时、分、秒，即 59、59、23 进位或清零，不必细说。只要再加上清零功能和开始暂停功能即可，开始暂停键可以统一，下图为模块代码：

```
module second_watch(en,signal,starp,clr,hour,minute,second);
   input en;input signal;input starp;input clr;
   output reg [4:0]hour = 5'b00000;
   output reg [5:0]minute = 6'b000000;
   output reg [5:0]second = 6'b000000;
   always @(posedge signal) begin
      if(en) begin
         if(starp && ~clr) begin
            if(second == 59) begin
               second <= 0;
               if(minute == 59) begin
                  minute <= 0;
                  if(hour == 23)
                     hour <= 0;
                  else hour <= hour + 1;
               end
               else minute <= minute + 1;
            end
            else
               second <= second + 1;
         end
         else if(clr) begin
            hour <= 0;
            minute <= 0;
            second <= 0;
         end
```

接下来需要在总模块中接入，将秒表对应的一套时间接入并输出，下图是总模块接口：

```
//second watch
second_watch s(sw,signal,starp,clr,hour_sw,minute_sw,second_sw);
```

闹钟模块：

就是和调整时间一样的逻辑，会有时钟信号，会有调整时、分、秒按键，输出一套时间

下图为模块代码：

```
module set_time(en,signal,adjust_h,adjust_m,adjust_s,hour,minute,second);
   input en;input signal;input adjust_h;input adjust_m;input adjust_s;
   output reg [4:0]hour = 5'b00000;
   output reg [5:0]minute = 6'b000000;
   output reg [5:0]second = 6'b000000;

   always @(posedge signal) begin
      if(en && adjust_h) begin
         if(hour == 23)
            hour <= 0;
         else hour <= hour + 1;
      end

      if(en && adjust_m) begin
         if(minute == 59)
            minute <= 0;
         else minute <= minute + 1;
      end

      if(en && adjust_s) begin
         if(second == 59)
            second <= 0;
         else second <= second + 1;
      end
   end
```

只需要在总模块中提供接口就行了，可以和调整时间共用一套调时、分、秒的按钮。当然输出的时间应该是闹钟的那一套时间。
下图为总模块中的接口：

```
//set_alarm
set_time st(alarm,signal,~adjust_h,~adjust_m,~adjust_s,hour_al,minute_al,second_al);
```

此外还需要最后一个模块，就是判断是否闹钟要响的模块，只要判断小时和分钟是否相等就行了，这样闹钟叫一分钟，比较符合常理
下图是模块代码，输出一个闹钟信号：

```
module is_equal(
   input signal,
   input [4:0]hour_true,
   input [5:0]minute_true,
   input [4:0]hour_al,
   input [5:0]minute_al,
   output reg bell = 0
);

   always @(posedge signal) begin
      if(hour_true == hour_al && minute_true == minute_al)
         bell <= 1;
      else bell <= 0;
   end

endmodule
```

在总模块中提供一个接口就行了

```
//decide whether alarm bell is ringing
is_equal ie(signal,hour,minute,hour_al,minute_al,bell);
```

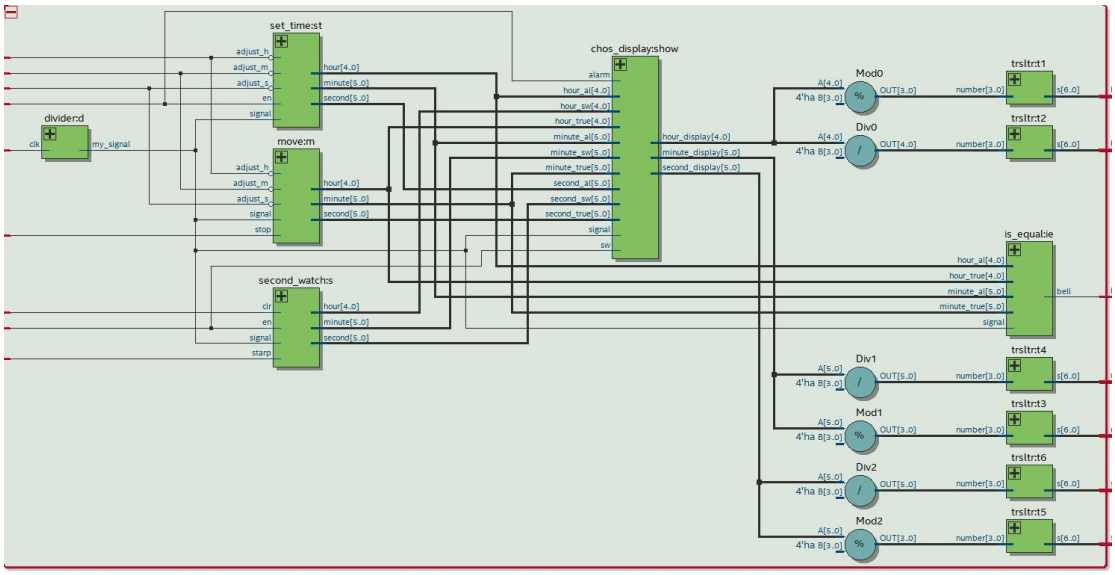
最后将数据选择器接入，即选择输出哪一套的时间，总模块中的选择输出接口：

```
//choose h.m.s to display
chos_display show(signal,sw,alarm,hour,minute,second,hour_sw,minute_sw,second_sw,hour_al,minute_al,second_al,
hour_display,minute_display,second_display);
```

最后将 hour_display,minute_display,second_display 输入到 7 段数码管就好了。
调用一下一开始的基础模块：

```
trsltr t1(.number(hour_display % 10),.s(hour2));
trsltr t2(.number(hour_display / 10),.s(hour1));
trsltr t3(.number(minute_display % 10),.s(minute2));
trsltr t4(.number(minute_display / 10),.s(minute1));
trsltr t5(.number(second_display % 10),.s(second2));
trsltr t6(.number(second_display / 10),.s(second1));
```

看下 RTL viewer 的模块图：



这样整个电子时钟就完成了！！！！

在顺便讲一下实验 5_1 的思路吧

要设计一个 0-99 的计数器，有开始、暂停、清零功能
用一个 7 位二进制数来存储时间 second，当 second == 99 或清零端有效时，second 置 0，开始、暂停键可以当作使能端输入，正常情况下 second ++,逻辑比较清晰，没有用很多的模块，下图是总模块代码：

```
input start;
input stop;
input clr;
output [6:0]s1;
output [6:0]s2;
output reg clk_1s = 0;
integer clk_count = 0;
reg [6:0]seconds = 7'b0000000;
always @(posedge clk) begin
   if(clk_count == 50000000) begin
      clk_count <= 0;
      clk_1s <= ~clk_1s;
      if(start && ~stop) begin
         if(clr || seconds == 99) begin
            seconds <= 7'b0000000;
            clk_count <= 0;
         end

         else begin
            seconds <= seconds + 1;
         end
      end
   end
   else
      clk_count <= clk_count + 1;
```

5.实验步骤 （第一部分和第二部分类似）

一．利用 systembuilder 建立工程

选择 CLOCK、SWITCH、LEDR、button

二．编写 Verilog 代码

写一下相应模块的代码，在顶层文件中做相应的调用，进行编译调试，直至成功。

三．在板子上进行测试

6.测试方法

在板子上进行各个功能的测试，没有用到 test_bench，注意进位情况，注意 button 按下是 0

7.实验结果

助教已验收。

8.实验中遇到的问题

1．一开始没有写分频器模块，导致后面很不方便，所以还是写了个分频器
2．一开始写电子时钟的时候，没有进行模块化编程，所有的功能几乎都缩在一个模块内，最后在实现秒表功能的时候，正常时钟无法工作并储存，这是不用模块化编程无法避免的问题，至少我没找到什么解决方法。。。所以最后就重构了一下思路，也学了下 verilog 模块实例化的部分，得到了现在的电子时钟。
3．SW[9]这个按键似乎有些问题，干脆不用他。
4．还有很多一开始不好解决的问题或者需要十分冗长的代码来实现，只要用了分模块编程，就很方便解决。

9.实验得到的启示

1．一定要学会用分模块编程，会让结构更加清晰，debug 更加容易
2．有问题可以向老师助教请教、和同学讨论、上网搜，很多时候是思路或结构出现了问题。

10.意见和建议

1．2000 多的板子费好大力气做一个手表，感觉有点怪，希望能做一些帅一点的。
2．这个实验好难，希望友好一点。。